Programmation et Exploitation des Platesformes HPC: Défis et challenges

François Bodin, Jean-François Méhaut

Programmation et exploitation des plates-formes HPC : Défis et challenges

François Bodin
Professeur, Université de Rennes 1, IRISA

Jean-François Méhaut
Professeur, Université de Grenoble Alpes, LIG

Introduction

La majorité des applications HPC repose sur des technologies et pratiques logicielles inventées il y a plusieurs décennies. Les considérations énergétiques, la fin de la loi de Moore et la production d'énormes volumes de données ont initié une mutation profonde du paysage du HPC où les équilibres calcul et stockage sont remis en cause. Face à cette rupture technologique c'est l'ensemble de la pile logicielle (systèmes, support d'exécutions, compilateurs, bibliothèques) et les codes applicatifs qui doit évoluer.

De l'impact de l'énergie sur le logiciel

La consommation énergétique est devenue le principal obstacle à la progression des performances en stoppant l'augmentation de la fréquence des processeurs. L'augmentation du nombre de cœurs s'est substituée pour fournir une puissance de calcul toujours accrue. Dans le même temps les technologies d'accélération (e.g., GPU, Xeon Phi, MPPA) ont aussi permis d'augmenter l'efficacité énergétique. En plus de cette massification du parallélisme, les nouvelles technologies de stockage (e.g., mémoire non volatiles, 3D *stacking*) et d'interconnexions (e.g., photonique) vont conduire à des architectures de machines, énergiquement efficace, mais de structures nouvelles et variées qui devront se refléter dans l'ensemble de couches logicielles. De plus, l'augmentation exponentielle du nombre de cœurs des supercalculateurs impose de mettre en place des schémas de tolérance aux fautes.

Prendre en compte l'ensemble de ces changements nécessite d'introduire des API (*Application Programming Interface*) capable de mesurer et de gérer la consommation d'énergie au niveau des applications. De plus, les modifications de la hiérarchie de stockage et les problèmes de fautes matérielles requièrent de nouvelles approches dans la gestion des entrées / sorties et de la mise en œuvre de la résilience (e.g., MPI tolérant aux fautes). Les nouvelles API devront permettre aux applications d'établir un dialogue de gestion des ressources matérielles performants et énergétiquement efficace.

Cette profonde évolution devrait conduire à une attribution des ressources utilisateurs, actuellement mesurée en heures.cpu, à une allocation en watt.heure.

Données extrêmes et HPC

La précision des simulations, les approches multi-physique ainsi que les techniques d'assimilation de données, parmi d'autres, produisent des volumes de données à gérer et à analyser tels qu'ils ont été caractérisé par la communauté de «déluge». En conséquence, l'exploitation et la gestion des données sont l'autre pan de l'évolution des logicielles qui va bouleverser les pratiques. Une majorité des applications HPC autrefois centrée sur une problématique de puissance de calcul sont maintenant fortement limitées par le traitement des données (incluant la visualisation). Les pratiques traditionnelles consistant à stocker sur disque l'ensemble des données produites pour les analyser a posteriori ne seront possibles que marginalement. L'analyse devra se faire « in-situ », en mémoire du calculateur. Naturellement, cette problématique a conduit à s'interroger sur l'introduction de technologies de type « Big-Data » dans le contexte HPC.

Du point de vue des couches basses logicielles, la cohabitation de techniques issues de l'analyse de données (data-mining ») impose, à terme, de faire cohabiter des technologies HPC traditionnelles avec celles du monde de l'analyse et traitement des données dans des *workflows* complexes. Un signe de cette évolution, les techniques de virtualisation, telle que les *containers*, font leur entrée dans les systèmes d'exploitation des supercalculateurs.

Hétérogénéité, performance et scalabilité

Les approches par décomposition de domaine et échanges de messages, fréquemment utilisées, peinent à prendre en compte les structures de machines fondées sur des processeurs massivement parallèles. L'hétérogénéité des ressources de calcul ajoute encore un degré de complexité. L'ensemble de la pile logicielle, en particulier le systèmes d'exploitation, les supports à l'exécution et le compilateur, doit permettre une gestion efficace, par exemple en supportant des paradigmes de programmation parallèle multi-niveau, scalable, permettant de mélanger des approches par mémoire distribuée et par mémoire partagée. Un point important est une régulation dynamique des charges de calcul qui prennent en compte les considérations énergétiques tout en respectant les affinités entre les tâches et les accès aux structures de données.

Par exemple s'agit de mélanger des approches par échange de message avec des approches basées sur des graphes de tâches capable de s'exécuter indifféremment sur les cœurs de CPU ou les éventuels accélérateurs (e.g., StarPU, OmpSS). Par ailleurs, les techniques d'optimisation automatique (e.g., auto-tuning) devront se généraliser tant

l'espace des possibilités à explorer est grand et, qui plus est, multi-critères (e.g., énergie vs performance, mémorisation vs re-calcul).

Complexité de la programmation

Pour être efficace, la programmation parallèle doit prendre en compte l'hétérogénéité des unités de calcul, la hiérarchie mémoire (mémoire cache et système de stockage) ainsi la hiérarchie du parallélisme matériel (SIMD, multi-coeurs, SIMT, cluster). La programmation hybride mixant des API telles que OpenMP, OmpSS pour le parallélisme intra-nœud et MPI ou encore PGAS (*Partionned Global Address Space*) entre nœuds ont fortement gagné en popularité ces dernières années. Cependant, l'obtention de code efficace reste un véritable challenge car la complexité croissante des machines nécessite la mise en œuvre de techniques d'optimisation de code élaborées et coûteuses en main d'œuvre.

Les approches à base de langages et API spécifiques à un domaine (*Domain Specific Language*) représentent un piste importante pour la séparation des aspects informatiques des aspects applicatifs. La définition de solution spécifique à un domaine est une recherche de compromis difficile entre simplicité de programmation et pérennité dans une domaine ou les standards sont long à émerger et nécessite une large communauté pour être économiquement viable. Les DSL a base de langages dynamiques, tel que Julia du MIT, présentent aussi un intérêt pour construire des environnements de programmation de très haut niveau (typiquement sur la base de méthodes de méta-programmation). Des progrès importants restent à faire pour aider à la mise au point des programmes massivement parallèles.

Conclusion

Les nombreux changements technologiques vont profondément altérer la façon de concevoir les applications. L'ensemble des couches logicielles seront revues pour prendre en compte les problématiques énergétiques, de gestion/exploitation de données et de tolérance aux fautes. Les pratiques de développement devront faire appel aux techniques de génie logiciel pour obtenir une architecture de code assez robuste pour s'adapter à plusieurs générations de machines. Les approches spécifiques à chaque domaine (DSL) permettront d'offrir des abstractions aux utilisateurs cependant il faudra trouver des modèles économiques permettant de servir une communauté donnée tout en assurant la pérennité des solutions. Les défis ne sont pas minces !

Références

- BDEC http://www.exascale.org/bdec/
- SRA ETP4HPC http://www.etp4hpc.eu/strategy/strategic-research-agenda/
- EESI Roadmap http://www.eesi-project.eu/
- PRACE http://www.prace-ri.eu/prace-the-scientific-case-for-hpc/